Chapter 04
Teaching Mobile App Development in a Mass Communication Program

by
Cindy Royal, Ph.D.

For the past two decades, I have taught web design and development courses to journalism and mass communication students. These courses began with covering Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and web animations and have now progressed to include responsive design techniques, front-end frameworks, and customizing content management systems. In 2014, I began teaching a more advanced web development course focused on interactive coding and data visualization. This course introduced JavaScript and charting libraries and presented students with techniques supporting interaction and user-experience design.

For years, I had avoided moving into the mobile application space. I had invested a lot of time learning web technologies. I believed in the open Internet and the ability to publish online without the constraint of distribution through application stores. I thought that the mobile application development environment introduced a level of complexity that was beyond the scope of our curriculum.

Eventually, I came around to the realization that it was time to introduce students to the ways in which mobile applications are made. The mobile application environment began to grow with the introduction of smartphones in the mid-2000s. In January 2018, 77% of Americans used a smartphone, but for young people aged 18-29, the percentage was 94% [3]. The total number of mobile application downloads in 2017 exceeded 178 billion and is expected to grow by 45% in 2020. [9]. The two most popular application stores, Google Play and the Apple App Store, housed 3.8 million and 2 million apps respectively in 2018 [8]. So, it became clear that mobile development had become an important part of any organization's communication strategy and needed a presence in our curriculum.

Mobile applications use different technologies than websites. These features include enhanced processing power and capabilities, including an accelerometer for detecting movement and global positioning systems that allow for location-based services [5], which are more difficult to execute on mobile devices with applications made for browser-based performance.

Two platforms have emerged as the leaders in the mobile environment: iOS and Android. The iOS platform is specific to the iPhone hardware, while the Android platform, developed by Google, works on several different devices made by Samsung, LG, and other manufacturers. The market for smartphones is competitive, with Apple having 44.6% of smartphone subscribers, Samsung at 29.7%, and LG at 9.6% on the Android platform.

Overall, Android has the most users of their smartphone operating system at 54.3% with Apple at 44.6% [12].

Mobile application development languages are becoming an in-demand competency [17]; [11], but few mass communication programs offer courses in these areas. Some programs have experimented with magazine-style applications using popular software programs like Adobe InDesign or InDesign-based plugins [4]. In 2013, the University of Texas School of Journalism and Mass Communication began teaching an app development course with journalism and computer science majors working collaboratively [6]. Even computer science departments have been slow to provide specific instruction on mobile application development, focusing instead on languages that support enterprise systems. Those who move into application development are often self-taught through online resources or enroll in programs in for-profit code camps, like those offered by General Assembly [1]; [7]; [10].

This chapter provides the process by which I developed and taught the Mobile Media Development course in the School of Journalism and Mass Communication at Texas State University. We had introduced a new degree in Digital Media Innovation in 2016, so the program was supportive of including emerging topics in curriculum.

Course Development Strategies

I chose to focus on the iOS and Apple development environment because I knew most of our students used iPhones and our labs were equipped with iMac computers. I set about learning the Xcode interactive development environment (IDE) and Swift programming language. Apple switched to Swift from Objective-C in 2014. Swift was introduced as a simpler, more powerful language to support the growing mobile development environment. I felt this was a good introductory platform for a mobile class.

I began my own training by reviewing Lynda.com courses starting with *Programming for Non-Programmers with iOS and Swift* [10] and using the free e-book from Apple on App Development with Swift [2]. While this was a new programming environment for me, I had experience with logic-based coding using JavaScript and Python, as well as experience with object-based programming environments when I had previously worked with Adobe Flash. However, for those new to programming, these tutorials start at a beginner level. It may be helpful, but not required, to have worked in Web development languages including HTML, CSS, and JavaScript. I have created a site at CodeActually.com that provides basic instruction and code samples on these topics. Seeking out other resources on sites like Codecademy.com can provide an introduction to programming terminology and processes that one might find helpful to understand before embarking on mobile application training.

In addition to the above resources, I spent a good deal of time searching the web for tutorials and code samples that would help me develop customized exercises that would provide meaningful context and support for communication students [13]. It was not too difficult to adapt to this new environment, but it did take a significant amount of time over two semesters to learn these skills and prepare lessons and exercises.

The objectives of the course were developed for students to be able to:

- explain the unique characteristics of the mobile communication environment

- evaluate the features and functions of an effective mobile application
- construct content and visuals for mobile distribution
- develop mobile applications that demonstrate the use of effective content, design, and functionality

The overall strategy for all my coding courses is to start with basic concepts and progressively build throughout the semester. Exercises take students through code samples and step-by-step examples of the types of features they will be required to demonstrate in their projects. Ideally, students watch tutorials before class, which we review in class and practice. I also spend a good deal of time explaining processes and features so students comprehend why and how things work and become comfortable with executing their own ideas and solving problems.

Course Segments, Exercises and Assignments

The eleven students taking the Mobile Media Development course in Spring 2018 had previously taken a Web Design course as a prerequisite, so while they were new to the mobile application environment, they had some experience making responsive websites. The Mobile course consists of three main production segments, each lasting about four weeks, using step-by-step coding tutorials in class and culminating with a project on a topic of each student’s choosing to demonstrate the specific concepts covered.

The course starts with modules on the mobile app development environment and the history of mobile. This includes a discussion of phones and mobile technologies, along with domestic and global usage statistics and demographics for the mobile environment. We discuss some of the categories of mobile applications, including social media, news, and information, mapping, messaging and email, weather, productivity, transportation, calendars and events, music and entertainment, ecommerce, and the emerging category of augmented reality. Subsequent modules deal with the features of devices that are unique to mobile development, including touchscreen gestures that provide different input methods for users - tapping, swiping, pinching and dragging.

During these modules, online discussions have students critique their favorite applications' relevant features and functionality and their own interaction with applications throughout the day. The lessons and discussion posts provide a foundation upon which to build the production elements of the course to support judgment and critical thinking.

Advanced Responsive Design

The first production segment of the course focuses on more advanced usage of responsive design in web applications to be served via browser on a mobile device. This allows the course to build upon concepts students learned in the web design course and introduce decision making on how to select which type of application development environment they should use. Conceptually, we discuss mobile development strategies, including the pros and cons of making responsive web applications versus mobile applications.

A series of exercises helps students apply techniques for percentage widths, media query breakpoints, and display options using CSS and creating responsive navigation. Finally, a brief introduction to HTML Forms, JavaScript and jQuery is provided, so students can implement simple interactivity on their sites. This topic also begins to broach conditional programming concepts, including if-statements and loops that will also be revisited in the mobile application segments.

The project that accompanies this segment is to create a mobile web app that demonstrates interactivity through form elements and manipulation of the Document Objective Model (DOM). Students can develop a calculator, a short quiz or other interactive project, as long as a user can interact with the page and be presented a result. It must be designed responsively and work well on a mobile browser.

The second segment of the course begins to introduce the iOS development environment. It starts with a general introduction to Swift programming concepts; use of variables and data types; concatenation; properties and methods and logical operations, if-statements; functions; loops; arrays; and objects **(See Figure 1)**. Since some of these concepts were introduced in the first segment related to JavaScript, this builds on prior knowledge and applies it to the new setting.

Programming Terminology

- **Variable -** A symbol or name that stands for a value.
- **Data Type -** Classification of a particular type of information – strings, numbers, etc.
- **Concatenation -** The act of linking together two or more objects.

- **Property -** A characteristic of an object. For example, the length property of a string indicates the number of characters.

- **Method -** A procedure that is executed when an object receives a message.

- **If Statement -** A programming decision based on meeting a condition or not.

- **Function -** A section of a program that performs a specific task.

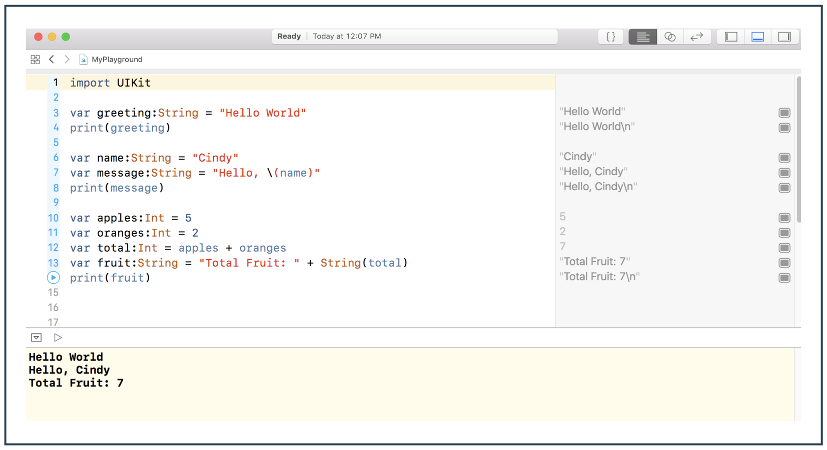
- **Loop -** A programming segment that repeats until a particular condition is met.

- **Array -** A simple list of elements.

- **Object -** A list of key-value pairs.

Figure 1: Programming Terminology

To gain practice with Swift, Xcode provides a Playground to try out code and learn language concepts. In Xcode, choose File:New:Playground, then choose the Blank template and Save. Students can run each line of code below individually, and the Playground will provide a response to the right of the code and console result in the bottom window **(Figure 2)**.



```swift
import UIKit

var greeting:String = "Hello World"
print(greeting)

var name:String = "Cindy"
var message:String = "Hello, \(name)"
print(message)

var apples:Int = 5
var oranges:Int = 2
var total:Int = apples + oranges
var fruit:String = "Total Fruit: " + String(total)
print(fruit)
```

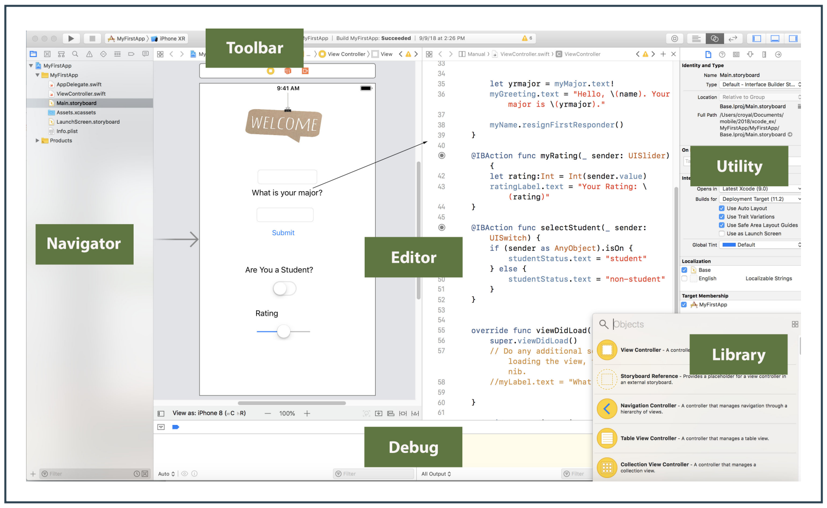
```
"Hello World"
"Hello World\n"

"Cindy"
"Hello, Cindy"
"Hello, Cindy\n"

5
2
7
"Total Fruit: 7"
"Total Fruit: 7\n"
```

```
Hello World
Hello, Cindy
Total Fruit: 7
```

Figure 2: Xcode Playground

Students are instructed to play with simple variable, math and concatenation concepts to become familiar with their execution in Swift. Swift code can also be run in the online Swift Playground at http://online.swiftplayground.run/.

Next, the Xcode interactive development environment (IDE) is introduced. It provides a combination of visual and coding windows and libraries of components. The first exercises demonstrate how to start a new project and use the sections of the interface. The Xcode interface can seem daunting at first—with multiple ways to show and hide windows, but once each section's purpose is described, students grow comfortable with how they will navigate it. The main sections we use are the Navigator to access various files, the Editor which has both coding and visual windows, and a Utility pane that changes depending on which element is selected. The Library pane provides access to resources provided by Xcode for creating controllers, buttons, and other features (In XCode 10, the Library is now a floating panel accessed by an icon on the top-right of the screen when the Storyboard is selected). The Toolbar allows the developer to run the application and see it in the Simulator. Errors can be addressed in the debugging area. See Figure 3 for an illustration of the Xcode 10 Interface.

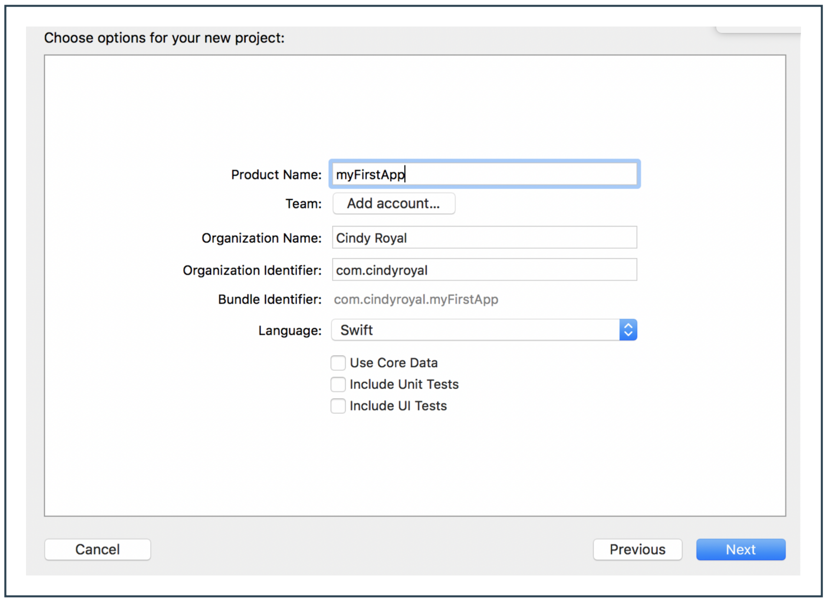


```
        let yrmajor = myMajor.text!
        myGreeting.text = "Hello, \(name). Your
            major is \(yrmajor)."

        myName.resignFirstResponder()
    }

    @IBAction func myRating(_ sender: UISlider)
        {
        let rating:Int = Int(sender.value)
        ratingLabel.text = "Your Rating: \
            (rating)"
    }

    @IBAction func selectStudent(_ sender:
        UISwitch) {
        if (sender as AnyObject).isOn {
            studentStatus.text = "student"
        } else {
            studentStatus.text = "non-student"
        }
    }


    override func viewDidLoad(
        super.viewDidLoad()
        // Do any additional s
            loading the view,
            nib.
        //myLabel.text = "What

    }
```

Figure 3: Xcode Interface sections

An Xcode project includes the .xcodeproj file and a folder including several supporting files. The Main.storyboard (visual interface) and the ViewController.swift (code) are the main files accessed. An Assets.xcassets folder is used to upload and store image and other resources used in the application.

When starting a new project, it is common to begin by using a Single View App template, although there are several other templates with which to begin a project. After the user provides a project name and other identifying information **(Figure 4)**, the project interface opens to a screen of options. To begin coding, one must use the Navigator area on the left to select a file.



Product Name: myFirstApp
Team: Add account...
Organization Name: Cindy Royal
Organization Identifier: com.cindyroyal
Bundle Identifier: com.cindyroyal.myFirstApp
Language: Swift
☐ Use Core Data
☐ Include Unit Tests
☐ Include UI Tests

Figure 4: Beginning an Xcode Project.

The default elements of the code interface provide the following statements and functions:

- import UIKit – imports the library of UI elements.
- class viewController – area of the code where most statements are made.
- func viewDidLoad method – statements in this function run when application loads.

The Storyboard and ViewController are accessed simultaneously in the Editor by opening one, then using Option-click to open the other. This is necessary to connect code to elements in the visual interface. Xcode provides a Simulator that you run from the Toolbar when you are ready to test your applications.

This segment of the course focuses on finding elements in the Xcode Library and applying outlets and actions. The basic operation within Xcode is to identify an element you want to use in the Library pane and drag it into the Storyboard window. This can be a Label or other UI element — Text Fields, Buttons, Switches, Sliders or more advanced features - depending on your desired usage. Once an item is selected in the Storyboard, by right-clicking (or ctrl-clicking), you can drag from the element into the viewController class (or other function) in the code window to begin coding. Dialog boxes appear to provide options depending on your desired functionality. This is the fundamental workflow in defining elements in the Xcode environment **(See code in Figure 5)**.

Creating Outlets
and Actions with

Xcode and Swift

An element can be turned into an Outlet or created as an Action. This is an important process to emphasize and practice with students. Turning a Label into an Outlet allows it to receive information programmatically. An Action on an element creates a function that gets executed on an event associated with the element.

This code applied to a Label element gives it a variable name that can be programmatically changed through an Action. By dragging from the Label to the code, a dialog box assists in providing the correct syntax. (In each case below, the IB stands for Interface Builder and is the required syntax in Swift to begin these statements.)

```
@IBOutlet weak var myLabel: UILabel!
```

This code creates an Outlet from a Text Field and gives it a variable name that can be referenced later in the program. In the same manner as the Label, by dragging from a Text Field to the code, a dialog box assists in providing the correct syntax.

```
@IBOutlet weak var myName: UITextField!
```

To create an Action, a Button that is dragged from the Library to the Storyboard can be used to assign a function that changes the value of a Label Outlet, as in welcoming a user to a site after completing a login form. In the same manner as above, by dragging from the Button on the Storyboard into the code, a dialog box assists with creating the correct syntax for the Action.

```
@IBAction func myButton(_ sender: Any)
{
     let name = myName.text!
     myLabel.text = "Hello \(name)"
}
```

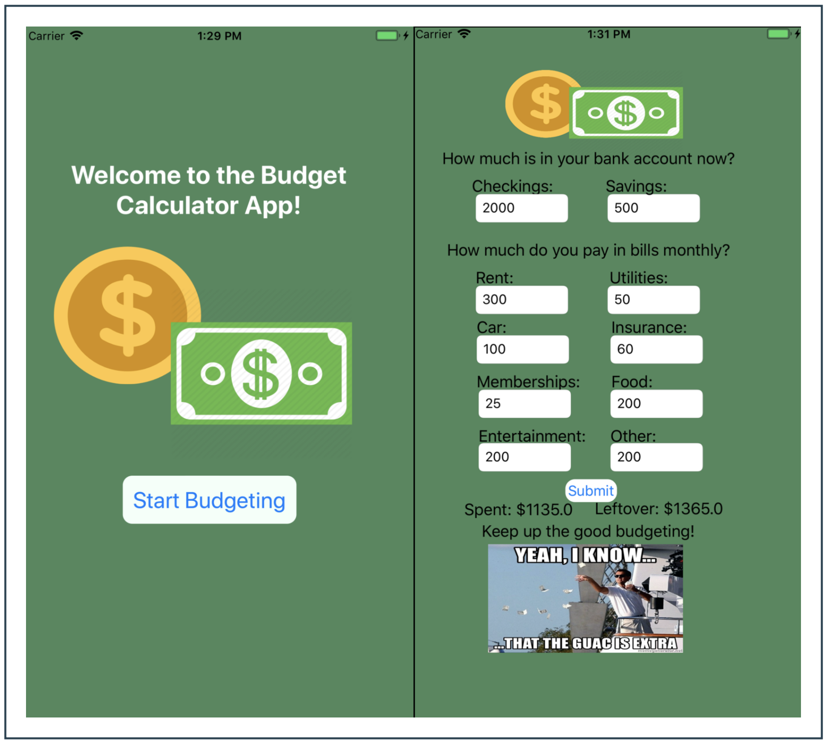
When an element in the Storyboard has a connection to the code, it is indicated by a filled circle instead of the line number. Connections can be edited in the Utility area with the Connections Inspector (small icon with arrow pointing to right).

The above code demonstrates a simple workflow for changing the value of a Label via user input and Button event. Exercises progressively work through several examples of the above process to get students familiar with manipulating Outlets and Actions, then introduce other input methods that include Switches, Sliders, and Picker Views.

Figure 5: Creating Outlets and Actions with Swift and Xcode

Subsequent exercises in this segment show options for including additional View Controllers (to function as additional pages or scenes in the app) and embedding a Navigation Controller to navigate through the application. While there are many more complex development and design techniques associated with mobile application development, these are the most basic concepts a student must grasp and will use throughout the rest of the course.

The project for this segment of the class is to create a simple quiz application. Students are to use Xcode and Swift to demonstrate interactivity via the use of Labels, Text Fields, Buttons, Switches or Sliders, and to develop an interface using Image Views, multiple View Controllers, and appropriate navigation. One example is this budgeting app created by a student in the class **(Figure 6)**.


Carrier
1:29 PM
Welcome to the Budget Calculator App!
Start Budgeting
Carrier
1:31 PM
YEAH, I KNOW...
...THAT THE GUAC IS EXTRA

How much is in your bank account now?

Checkings: 2000
Savings: 500

How much do you pay in bills monthly?

Rent: 300
Utilities: 50
Car: 100
Insurance: 60
Memberships: 25
Food: 200
Entertainment: 200
Other: 200

Submit

Spent: $1135.0 Leftover: $1365.0

Keep up the good budgeting!

Figure 6: Budgeting App Developed by Danielle Molinar in Mobile Media Design course.

The third segment of the course deals with advanced application functionality. At this stage, students have become familiar with the process for creating Outlets and Actions and have gained experience with some basic input methods and functionality. An exercise **(Figure 7)** is used to demonstrate simple math operations and functions in a game application.

Other exercises in the course show students how to pass data between multiple View Controllers and how to implement additional user input methods. Exercises provide examples for how to select images from the camera roll, implement maps, add web links to an application and use data in an application.

Demonstrating Math Operations and Functions

The game exercise demonstrates how to create two buttons, one that increments a score by one point and the other that increments it by five points. The functions myAlien and myPredator are attached to the buttons. The code below demonstrates how to create the function didScore that increments the score with points passed to it from the button functions.

```
func didScore(points:Int){
    score = score + points
    myScore.text = "Score: \(score)”
}

@IBAction func myAlien(_ sender: Any) {
    didScore(points:1)
}

@IBAction func myPredator(_ sender:
Any) {
    didScore(points:5)
}
```

Once this functionality is established, students discuss ways that the game can be made more challenging. One way is to create the code that resets the score to 0. The exercise goes through several different ways to create a “chars” variable to be used in a modulo function to reset the score to 0 if the score is divisible by a number. A random number is generated in a “chars” variable (the code below creates a random number between 2-12), and the didScore function is modified with an if statement to reset the score to zero when the score is divided evenly (without a remainder) by the number.

```
var chars:Int =
Int(arc4random_uniform(10) + 2)

func didScore(points:Int){
    score = score + points
    if(score % chars == 0) {
    score=0
}
myScore.text = "Score: \(score)"
}
```

The interface for this project is a simple layout of two buttons and a label to hold the score. The focus is on functionality. The goal of the game is to get as many points as possible without resetting the score, so the player must figure out the pattern. A timer function is later added to the game. The full exercise can be found at mobile.cindyroyal.net.

This simple math exercise provides the foundation for students to be able to add more complex formulas to their projects. Giving students step-by-step instructions, this method allows them to think through more complex processes using proper coding standards to later apply on their own.

Figure 7: Game Application Code

A module on Layout features in Xcode is introduced. These are Constraint and Alignment features that allow for the interface to be flexible to work on devices of various sizes. The Auto Layout Toolbar is found in the bottom right corner of the Storyboard window. The buttons are used to update frames, stack and embed elements, add alignment constraints, and resolve layout conflicts **(see Figure 8)**. Although students in this course are primarily focused on creating functioning code, attention to these design implications is recommended.

Figure 8: Xcode Auto Layout Toolbar and Add New Constraints Interface

The final modules of the course demonstrate how to test a project on a device and the process of submitting an application to Apple's App Store for distribution.

We also discuss using design-thinking processes and prototyping tools, before spending time coding. While the Xcode interface is relatively simple, it is helpful to have a general idea of how one wants to design an application, either by using pencil and paper or a prototyping software program, before beginning to code. Other concepts we discuss in the course include the programming repository GitHub. Students use GitHub.com to upload their project files so they can be graded. GitHub is a type of social network for developers that allows for code sharing and collaboration. Students are also introduced to the agile development methodology, so they are familiar with iteration and collaboration processes in the software development industry [16].

Student Work

The final project in the course is a demonstration of all the app development features to which the student has been introduced throughout the semester. The project is designed to allow students to exercise creativity in selecting their topic and functionality, and challenges them to implement advanced features. One student developed an application that asked the user to unscramble sections of a photo by swapping pieces. Another student created a timed math quiz with three levels of difficulty **(Figure 9)**.

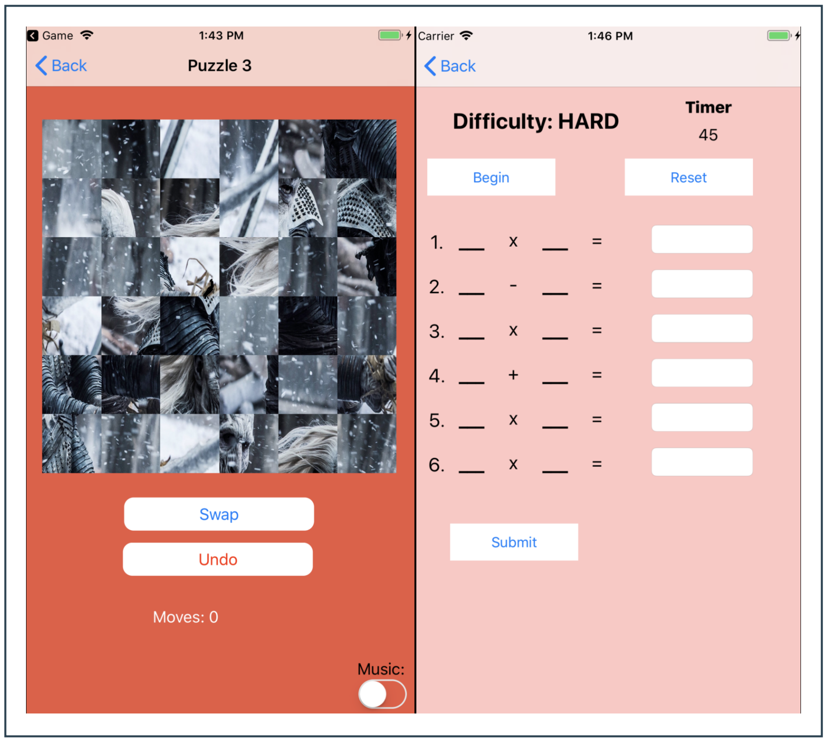


Figure 9: Screen captures from two final projects: an image-swapping app by Danielle Molinar and a timed-math quiz by Sydney Dorsey.

Future Course Considerations

Different students learn at different rates and through different means. These are difficult concepts for all to grasp, and even the best students are still app development novices at the end of the course. Exercises are developed to allow all students to demonstrate basic functionality in their projects and work at their own speed, but reward students who put in extra effort and creativity in bringing their application ideas to reality.

Throughout the course, I emphasize problem-solving and troubleshooting. It is important for students in any coding course to understand that the practice of programming is one in which solving problems is the norm and that programs often don't work on first execution. I emphasize confidence building through small successes leading to more advanced logic and functionality and encourage students to have fun with the creative process in which they are participating.

As I began teaching the Mobile Media Development course again in 2019, I have incorporated a few modifications. I reduced the responsive web application segment of the course to a shorter exercise and started more quickly with mobile application development with Xcode and Swift. I felt that these are complex topics that could use additional time and that a more advanced course in responsive design and user experience would better suit those topics.

With the additional time, I extended the Layout and Design segment to provide a stronger appreciation of how to implement the Alignment and Constraint features in Xcode. The final course segments now include cloud database integration with Google's Firebase and descriptions of the Android development environment and cross-platform development with frameworks like the JavaScript-based React Native. In the future, I'd like to experiment with the Augmented Reality features provided in Apple's ARKit components.

As the course evolves, I'd like to incorporate more attention to conceptual issues to include modules on information security and user privacy concerns. The applications created in this course do not share user data. However, addressing these topics will provide students with critical thinking and judgment skills to apply to future projects. My course site hosts my lessons and tutorials, and I plan to continue to improve the presentation of code samples on the site (mobile.cindyroyal.net).

With the high percentage of female students in journalism and mass communication programs, by teaching coding, we have a unique opportunity to introduce technology skills to those under-represented in computer science and other technology-based majors [14], [15] . As in other coding courses in mass communication programs, the goal is to not necessarily create fully formed professional developers. However, the insight into how mobile applications are made provides students with a strong perspective to bring to their careers as they make recommendations and work on collaborations that are likely to include mobile strategies. Learning coding demystifies the technology development process, builds problem-solving skills and confidence and can be a lot of fun! These skills will make students more desirable for a range of careers, and their presence in mass communication curriculum will continue to be in demand.

Chapter 4 Citations

To cite this article:

MLA: Royal, Cindy. "Teaching Mobile App Development in a Mass Communication Program." *Coding Pedagogy*, edited by Jeremy Sarachan, 2019, ch. 4, http://codingpedagogy.net. Accessed 1 Apr. 2020. *[update access date]*

APA: Royal, C. (2019). "Teaching Mobile App Development in a Mass Communication Program." In J. Sarachan (Ed.), Coding Pedagogy, ch. 4. Retrieved from http://codingpedagogy.net.

Chicago: Royal, Cindy, "Teaching Mobile App Development in a Mass Communication Program," in *Coding Pedagogy*, ed. Jeremy Sarachan, ch. 4, *Coding Pedagogy*, 2019. http://codingpedagogy.net.

"10 Best Online Courses for Learning Mobile App Development" (2018). Justinmind. https://www.justinmind.com/blog/learn-mobile-app-development-with-these-10-online-courses/.

“App Development with Swift by Apple Education on Apple Books.” Apple Books. https://itunes.apple.com/us/book/app-development-with-swift/id1219117996?mt=11.

"Demographics of Mobile Device Ownership and Adoption in the United States" (5 Feb. 2018). Pew Research Center. http://www.pewinternet.org/fact-sheet/mobile/.

Fletcher, Carol (2011). "Going Mobile with Student Magazines." Journal of Magazine & New Media Research, 12(2), pp. 1-9. https://aejmcmagazine.arizona.edu/Journal/Summer2011/Fletcher.pdf

Holzer, Adrian, and Jan Ondrus (Feb. 2011). “Mobile Application Market: A Developer’s Perspective.” Telematics and Informatics, 28(1), pp. 22–31. ScienceDirect, https://www.janondrus.com/pubs/mobile-application-market-a-developers-perspective.

“Journalism, Computer Science Collaborate in Mobile Apps Class” (22 Sept. 2015). School of Journalism, Moody College. https://journalism.utexas.edu/news/journalism-computer-science-collaborate-mobile-apps-class.

Liu, Trista (11 Feb. 2018). “10 Best APP Development Courses for Beginners and Get a Job.” Medium. https://medium.com/@tristaljing/10-best-app-development-courses-for-beginners-and-get-a-job-d84dbf34b101.

Loesche, Dyfed (2018). “The Biggest App Stores.” Statista. https://www.statista.com/chart/12455/number-of-apps-available-in-leading-app-stores/.

"Number of Mobile App Downloads Worldwide in 2017, 2018 and 2022" (2018). Statista. https://www.statista.com/statistics/266488/forecast-of-mobile-app-downloads/.

Perkins, Todd. “Learn Mobile App Development.” Lynda.Com. https://www.lynda.com/iOS-tutorials/Programming-Non-Programmers-iOS-11-Swift/642473-2.html.

Petkov, Alexander (16 Jan. 2018). “Here Are the Best Programming Languages to Learn in 2018.” FreeCodeCamp.Org. https://medium.freecodecamp.org/best-programming-languages-to-learn-in-2018-ultimate-guide-bfc93e615b35.

“Rankings" (June 2018). Comscore, Inc.. http://www.comscore.com/Insights/Rankings.

Royal, Cindy (2017). "Coding the Curriculum: Journalism Education for the Digital Age." Global Journalism Education in the 21st Century: Challenges & Innovations. Knight Center for Journalism in the Americas. https://knightcenter.utexas.edu/books/GlobalJournalism.pdf.

Royal, Cindy (2012). "Journalism Schools Need to Get Better at Teaching Tech Where the Girls Are. Nieman Journalism Lab." http://www.niemanlab.org/2012/10/cindy-royal-journalism-schools-need-to-get-better-at-teaching-tech-where-the-girls-are/.

Royal, Cindy (Dec. 2017). "Why Texas State’s Digital Media Degree Is Attracting Female Students." Mediashift. http://mediashift.org/2017/12/digital-media-degree-attracting-female-students/.

Sacolick, Issac (Mar. 2018). “What Is Agile Methodology? Modern Software Development Explained | InfoWorld.” Infoworld. https://www.infoworld.com/article/3237508/agile-development/what-is-agile-methodology-modern-software-development-explained.html.

Sandler, Rachel (20 Apr. 2018). "14 Most Popular Programming Languages According to Stack Overflow Study." Business Insider. https://www.businessinsider.com/14-most-popular-programming-languages-stack-overflow-developer-survey-2018-4.